UNIVERSIDAD CARLOS III DE MADRID working papers

Working Paper 11-09 (06)
Statistics and Econometrics Series
April 2011

Departamento de Estadística
Universidad Carlos III de Madrid
Calle Madrid, 126
28903 Getafe (Spain)
Fax (34) 91 624-98-49

A VEHICLE ROUTING MODEL WITH SPLIT DELIVERY AND STOP NODES

Leonardo Berbotto[1], Sergio García[1] , Francisco J. Nogales [1]

Abstract

In this work, a new variant of the Capacitated Vehicle Routing Problem (CVRP) is presented where the vehicles cannot perform any route leg longer than a given length L (although the routes can be longer). Thus, once a route leg length is close to L, the vehicle must go to a *stop node* to end the leg or return to the depot. We introduce this condition in a variation of the CVRP, the Split Delivery Vehicle Routing Problem, where multiple visits to a customer by different vehicles are allowed. We present two formulations for this problem which we call Split Delivery Vehicle Routing Problem with Stop Nodes: a vehicle flow formulation and a commodity flow formulation. Because of the complexity of this problem, a heuristic approach is developed. We compare its performance with and without the stop nodes.

Keywords: Split Delivery Vehicle Routing Problem, Stop Node, Granular neighborhood, Tabu search.

Acknowledgements: The first and third authors are supported by the Spanish Government through project MTM2010-16519. The second author is supported by the Spanish Government through project MTM2009-14039-C06-04.

[1] Department of Statistics, Universidad Carlos III de Madrid.
e-mail addresses: lberbott@est-econ.uc3m.es (Leonardo Berbotto),
sergio.garcia@uc3m.es (Sergio García),
FcoJavier.Nogales@uc3m.es (Francisco J. Nogales).

A Vehicle Routing Model with Split Delivery and Stop Nodes

Leonardo Berbotto, Sergio García, Francisco J. Nogales
Department of Statistics
Universidad Carlos III de Madrid, Spain
lberbott@est-econ.uc3m.es, sergio.garcia@uc3m.es, FcoJavier.Nogales@uc3m.es

Abstract

In this work, a new variant of the Capacitated Vehicle Routing Problem (CVRP) is presented where the vehicles cannot perform any route leg longer than a given length L (although the routes can be longer). Thus, once a route leg length is close to L, the vehicle must go to a *stop node* to end the leg or return to the depot. We introduce this condition in a variation of the CVRP, the Split Delivery Vehicle Routing Problem, where multiple visits to a customer by different vehicles are allowed. We present two formulations for this problem which we call Split Delivery Vehicle Routing Problem with Stop Nodes: a vehicle flow formulation and a commodity flow formulation. Because of the complexity of this problem, a heuristic approach is developed. We compare its performance with and without the stop nodes.

Keywords: Split Delivery Vehicle Routing Problem, Stop Node, Granular neighborhood, Tabu search.

1. Introduction

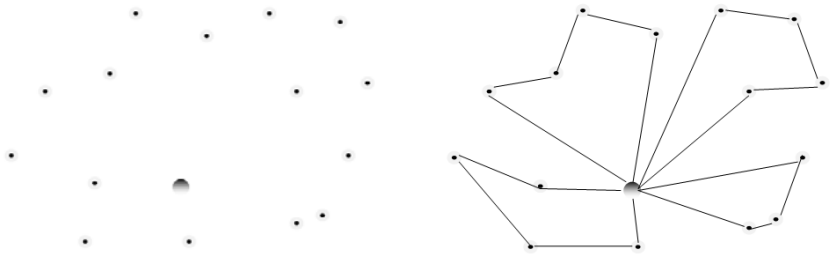
Transportation problems are confronted almost every day by everyone. From the simplest one, like organizing a shopping day, to the most complex one, faced up by logistic companies, both need a "good" model such that it captures the characteristics of the problem in the best possible way. Thus, routing problems are among the most studied problems in Combinatorial Optimization. The so-called *Capacitated Vehicle Routing Problem* (CVRP) belongs to this family of problems: a set of routes for a fleet of capacitated vehicles, based at one or several depots, must be determined for a number of geographically dispersed cities or customers. Each of them has to be visited by one single vehicle. Figure 1 shows an example of a CVRP: in Figure 1a, there is a set of customers with some demands. The central point is the depot. Figure 1b shows a routing solution with four vehicles serving the customers. The objective is to serve the customers, whose demands are known in advance, with all routes beginning

[1]The first and third authors are supported by the Spanish Government through project MTM2010-16519. The second author is supported by the Spanish Government through project MTM2009-14039-C06-04.

and ending at a depot such that the total cost (total length of the routes) is minimum. Other usual objectives for the CVRP are the minimization of the global transportation cost or the travel time, the minimization of the number of vehicles used, the minimization of the penalties associated with partial services if the customers are not fully served, etc.

Figure 1: Vehicle Routing Problem

(a) Set of customers and a depot (b) Solution of VRP



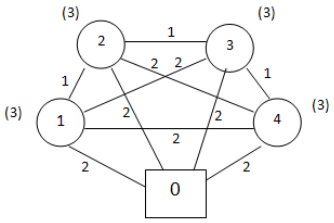
Real life situations need to adapt the CVRP model to their own characteristics and, thus, new variants of the CVRP are obtained in a natural way. An important application of the CVRP, not studied until now, is to consider a maximum length on each route leg performed by a vehicle. First, we define a route leg as a piece of route starting and ending at the depot or at some special node different from a customer. Thus, a route may be divided in more than one leg with lengths less than an specific measure each. A simple example is the following: a route length is equal to 720 km but each route leg cannot be longer than 400 km. This route may be divided in different ways: one route of 400 km and other one with 320 km or both route legs with 360 km each, or any combinations of route legs less than 400 km each. Transportation examples that may consider this variant of the CVRP are those situations where the driver cannot work more than a certain amount of time without having some rest or where a vehicle must find a gas station before a certain limit of kilometers. This new condition is added to the CVRP model such that, once the vehicle is close to the maximum length, it must return to the depot or find a place to stop before the route leg length limit is exceed. The places where the vehicle can stop, which we assume to be different from the set of customers, are called *stop nodes*.

Other variant of the CVRP already known in the literature is the *Split Delivery Vehicle Routing Problem* (SDVRP). It is a relaxation of the CVRP that removes the constraint of each customer being served by just one single vehicle. We are also interested in this variant of routing problems because it has many real life applications and, in some cases, savings with respect to the transportation cost of the CVRP are reached. This fact is illustrated in Figure 2. Figure 2a shows the graph for a simple problem with its arc costs. The customer

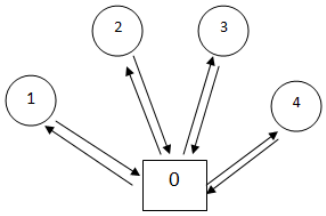
demands are equal to 3 for all of them and the vehicles have capacity $Q = 4$. Figure 2b shows the optimal solution of the CVRP: four vehicles are used and the cost is 16. If split delivery is allowed, Figure 2c shows an optimal solution: three vehicles are needed and the solution cost is 15.

Figure 2: Split Delivery Vehicle Routing Problem

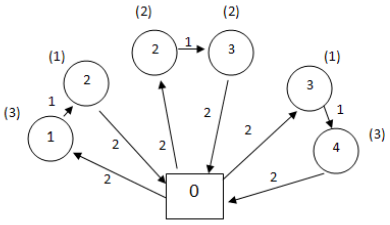
(a) Set of customers and a depot



(b) Solution without split



(c) Solution with split



A first study on this saving was carried out in [19] using a heuristic approach. For problem sets of 75, 115 and 150 demand points and a fleet of vehicles with capacity fixed at $Q = 160$ units, the average number of vehicles saved was 14.07 and the average percentage distance gained for this set of problems was 11.24% for demands between 0.7Q and 0.9Q. They also showed empirically that if the demands are less than $0.1Q$, there is no (or very little) advantage in solving the problem as a SDVRP versus a CVRP. These savings are theoretically studied in [6]: the cost of an optimal CVRP solution is compared with the cost of an optimal SDVRP solution by computing an upper bound on the ratio between the solution of both problems. It is shown that savings can be at most 50% and that this bound is tight (i.e., there exist instances where the value of the optimal solution of the CVRP is exactly twice the optimal value of the SDVRP).

The SDVRP has been applied successfully to several real situations such as the management of a fleet of trucks in a feed distribution problem ([33]), the problem of determining the flight schedule for helicopters to off-shore platforms for exchanging crew people employed on these platforms ([34]), and a waste collection problem with vehicles with small capacities and customers with demands greater than the vehicle capacity ([7]). In [10], a scatter search approach is applied to a retail composed by 519 stores in 11 Brazilian states.

For these reasons, a model that combines split delivery and constraints on the route leg length seems an interesting transportation problem, to study. This work presents a first model for the Split Delivery Vehicle Routing Problem with Stop Nodes (SDVRPSN) using two formulations: a vehicle flow formulation and a commodity flow formulation. For solving the SDVRPSN, we develop a tabu search heuristic that we call Hybrid Granular Tabu Search (HGTS). The local search is done in a granular neighborhood as developed in [35] for routing problems. The heuristic uses some move operators dealing with the routing problem and others that focus on the split problems and on the route length. Since we allow to search in neighborhoods with infeasible solutions with regard to the vehicle capacity and route leg length, the algorithm introduces corrective phases to fix these excesses. Because there are not benchmarks for the SDVRPSN, we solve the SDVRP using HGTS and the same instances than [8]. We compare the results with the best heuristics found in the literature and thus we can test the algorithm efficiency. The HGTS performs very well and outperforms many well-known solutions of the tested instances. After this, the limit on the route leg length is introduced by fixing it to twice the distance between the depot and the farthest customer. The problem is solved with HGTS for a subset of the instances that present the lowest demand with respect to the vehicle capacity. We select these instances because their solution have a high probability of presenting more than one route leg on each route.

The rest of this paper is structured as follows: Section 2 reviews the literature for the SDVRP and for problems with some links to this new variant (SDVRPSN). Section 3 introduces the general framework of routing problem with route leg length constraints and Section 4 presents this variant specifically for the SDVRP. Section 5 describes the HGTS and computational results are exposed in Section 6. Finally, some conclusions and further research are given in Section 7.

2. Literature review

The CVRP was introduced in 1959, when the first mathematical programming formulation and the first algorithmic approach were proposed in a real world application about gasoline delivery (see [16]). These initial results were later improved in [14], where it was developed a greedy heuristic, and which has become one of the most classical papers on the topic. The CVRP falls into the category of NP-hard problems, which means that the computational effort required to solve the problem increases exponentially as the problem size grows linearly. As a consequence of its complexity, most of real-life instances can only be solved with heuristics.

The variant of the CVRP with split delivery was introduced in [19] and [20], where it was shown the potential in cost saving through split deliveries. This model is more realistic than the simple CVRP in, for example, waste collection or food distribution, situations where the demand at the nodes is much larger than the capacities of the vehicles. SDVRP is a more complex problem than CVRP because it has many more feasible solutions. Some valid inequalities for the SDVRP are derived in [18] and it is shown in [9] that the dimension of the SDVRP polyhedron depends on whether a vehicle visiting a customer must service or not at least one unit of the customer demand. They propose a new family of valid inequalities that define facets of the polyhedron and a cutting plane algorithm whose quality is exhibited by solving instances with up to 50 customers (but individual customer demands do not exceed the vehicle capacity). In [30], it is presented a column generation approach similar to the one in [34] and 12 of the 25 instances used in [9] (those with customer demands at least 15% larger than the vehicle capacity) are compared. They improve the error gap (difference between lower bound and upper bound generated by the same column generation algorithm) with respect to the results of [9].

Like the classical CVRP, the SDVRP is NP-hard ([20]). As a consequence, exact solution methods are few and cannot solve problems too large in size. In [32], the problem is formulated as a dynamic programming model and instances with up to nine customers are solved. In [29], a two-stage algorithm is presented: i) the first stage creates clusters that cover all the demands and establishes a lower bound for the optimal value; ii) the second stage calculates the minimal distance traveled for each cluster by solving the corresponding traveling salesman problem and also establishes an upper bound. This approach is able to solve instances with up to 23 customers using a large computational time (more than 13 hours). Finally, an exact algorithm for the SDVRP with Time Windows (where customers must be visited within a known time range) based on a set covering formulation and a column generation technique is proposed in [24]. The algorithm solves almost all instances with up to 50 customers. In [17], a branch-and-price-and-cut method for this variant of SDVRP is proposed. It tests 504 instances, among which 176 instances are solved to optimality within one hour of computer time.

Because of the already mentioned complexity of the problem, heuristic procedures are more often found in the literature. In [19] and [20], a first local search approach is presented. Tabu search heuristics are used in [2], [8] and [28]. In [28] the authors compare the saving between solutions with and without split delivery in a model with time windows. They use the Solomon instances (see http://www2.imm.dtu.dk/ jla/solomon.html) to show that the number of vehicles and the distances traveled are reduced when split deliveries are allowed. For other cases where split deliveries are not efficient, they obtain results identical to the best

known solution with no splits. A very simple tabu search with only two procedures named Order Routes and Best Neighbor is implemented in [8]. They also add an improving phase using the GENIUS algorithm developed in [25] and a *k-split cycle* elimination procedure. They compare their results with [19] and improve almost all the considered instances. Finally, in [2] a tabu search and a learning procedure are applied. The method is based on an set of initial solutions that are used to build a new solution with high quality by replacing the existing solution with least quality. The generated solutions are improved with a variable neighborhood descendent procedure presented in [4].

On more solution algorithm for the SDVRP can be found in [12], where the first algorithm based on a scatter search methodology for this problem is presented. It works with the minimum number of vehicles necessary to serve all the demands. In [11], a memetic algorithm with population management is implemented, which combines a genetic algorithm with local search for intensification and diversification.

Following with heuristics for the SDVRP, the tabu search procedure of [8] is used in [5] to identify part of the solution space that has a high probability of being in a best solution. After that identification, an integer program is run to obtain improved feasible solutions. Moreover, a hybrid approach combining a mixed integer program and a record-to-record travel algorithm producing high quality solutions is developed in [13]. In [3], where a review on solution techniques for the SDVRP can be found, a new diversification methodology is developed. The geographic space of the problem is marked with rings used to group the costumers and then they are assigned to routes using a constructive approach.

Concerning with models that introduce conditions on the route length, we can find in [31] a limit on the total length of a route but not on route legs. Moreover, close to the idea of the stop nodes, in [1] a model is proposed where routes must end at the driver's homes or at a parking lot. Therefore, even though the model with stop nodes is a realistic situation and it is present in many situations, it has not been studied yet. We introduce this new variant by describing the main characteristic of the route leg length constraints and the definition of the stop nodes.

3. The CVRP with Stop Nodes

In this section we present the new set of conditions on the route leg length that added to the CVRP model define the CVRP with Stop Nodes (CVRPSN). This approach differs from others that introduce the limit length on the complete route, because in these problems

the minimum number of vehicles needed to serve all the customers may be greater than that in the new variant. In a model with a limit on the route length, the minimum number of vehicles needed to solve the problem does not depend on just the customer demands and on the vehicle capacity, but also on the distances. Also, in a model with route length constraint, the limit value of this length must be an amount at least twice the distance of the farthest customer to the depot for some feasible solution to exist. However, with the use of stop nodes, a lower value of the limit on route leg length does not exist because to reach a customer you can visit many stops as needed. Then, the minimum number of vehicles depends just on the customer demands and on the vehicle capacity. Of course, we need a necessary amount of well located stop nodes.

3.1. Motivating example

For a better explanation about the use of stop nodes, consider the case where the vehicle cannot perform a route leg longer than a given length L. First of all, we introduce in the problem a new component called *stop node*: a node with no demand where the vehicle can stop when the route leg length is close to L. This means that the cumulative length along the route leg must be computed such that it cannot exceed the given value L. Thus, when the route leg length is close to this limit, the vehicle must either return to the depot or go to a stop node. If the latter choice is taken, after the stop, the vehicle can resume the route and the leg length starts with a cumulative length equal to zero. The value of L can be measured in distance units (i.e., kilometers) or in time (i.e., hours). In the problem that we study here that we assume the existence of a set of stop nodes, different from the customer set, with known location and without demands.

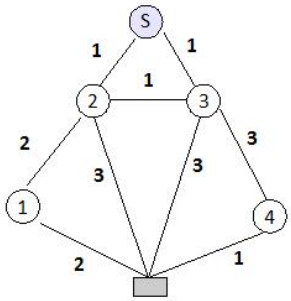
Figure 3 shows an example of CVRPSN. Assume that there is a set of 4 customers $C = \{1, 2, 3, 4\}$, with one unit of demand each, a depot 0 and a stop node s (Figure 3a). Suppose that a route leg cannot exceed a maximum length $L = 7$; the vehicle capacity is $Q = 3$ and the number of vehicles is $K = 2$. The objective is to minimize the total route cost of serving all the customers. For the sake of simplicity, we assume that the lengths of the arcs are symmetric ($l_{ij} = l_{ji} \ \ \forall i, j \in C \cup \{0\} \cup \{s\}$) and that these lengths are equal to the travel cost along each arc ($c_{ij} = l_{ij} \ \ \forall i, j \in C \cup \{0\} \cup \{s\}$). We assume the following set of costs:

- $c_{23} = c_{04} = 1, \ \ c_{01} = c_{12} = 2, \ \ c_{02} = c_{03} = c_{34} = 3$;
- the cost of the arcs between the stop node s and the nodes 2 and 3 are equal to one, that is $c_{2s} = c_{3s} = 1$;
- the remaining costs are greater than 3.

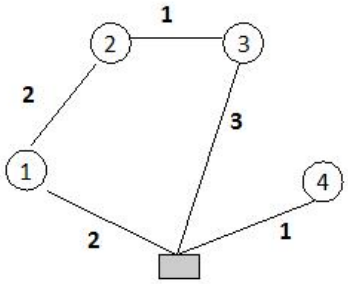
We assume a cost of one unit when the stop node is used. First, consider the case where there is no stop node. The solution of the CVRP without the route leg length constraint is equal to 10 (Figure 3b). If this constraint is introduced but the use of the stop node is forbidden the solution cost is 14, performed in two routes: vehicle 1 visits nodes 1 and 2 and vehicle 2 visits the nodes 3 and 4. Both route lengths are equal to 7 ($c_{01} + c_{12} + c_{20} = 7$ and $c_{03} + c_{34} + c_{40} = 7$). This is shown in Figure 3c.

Figure 3: SDVRPSN with maximum length equal to 7

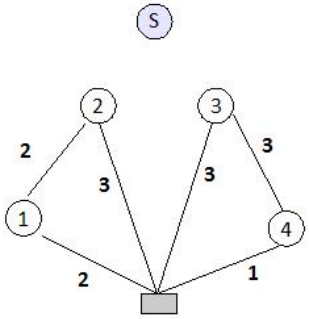
(a) Set of customers, depot and stop node



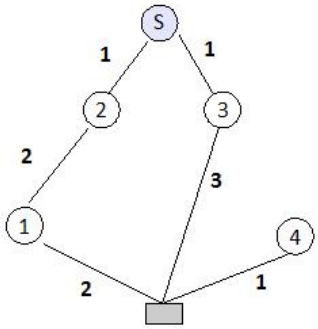
(b) Solution of CVRP with no L ($z = 10$)



(c) Solution of CVRP with L and no stop node ($z = 14$)



(d) Solution of CVRP with L and stop node ($z = 12$)



Now, consider that the use of the stop node is allowed. This means that a vehicle might go to the stop node if the route leg length is close to L. The optimal solution is also performed in two routes, but this time with a cost of 12. In the first tour, vehicle 1 visits nodes 1 and 2, the stop node s and finally node 3. Note that, at node 2 the route leg length is 4 ($c_{01} + c_{12} = 4$). If it decides to go to node 3 and then from node 3 to the depot, then the route leg length will exceed L. If the vehicle goes to the stop node, the route leg length is 5 ($< L$). Then, it resumes the next route leg from the stop node and, because the remaining capacity on the vehicle is 1, the last visited customer on this route is node 3 before returning to the depot with a leg length equal to 4 ($c_{s3} + c_{30}$). This route has two legs, both with length less than L. Vehicle 2 visits the fourth customer with a route length of 2 (Figure 3d). A cost of 1 must

be added to the objective because the use of the stop node by vehicle 1.

The idea of this graphical example is formalized next with some constraints.

3.2. Route leg length constraints

Route leg length constraints forbid the existence of a route leg length larger than a given value L. Let l_{ij} represent the length of the arc from node i to node j. Because we have this new kind of length constraints, the computation of the cumulative length at each node is necessary, meaning that we need to calculate the leg length performed by each vehicle by every node that it visits. We denote with a_{jk} the cumulative route leg length when vehicle k arrives at customer j. For example, in Figure 3d, where the order of the customers on route 1 is $0-1-2-s-3-0$, the computation of a_{j1} at each node is: $a_{11} = l_{01} = 2$, $a_{21} = a_{11} + l_{12} = 4$, $a_{s1} = a_{21} + l_{2s} = 5$, $a_{31} = l_{s3} = 1$ and, finally, $a_{01} = a_{31} + l_{30} = 4$.

The formulation of these constraints is the following. Let $C = \{1, 2, \ldots, n\}$ be the set of customers, each one with a positive demand d_i and let $S = \{n+1, n+2, \ldots, n+p\}$ be the set of stop nodes (which have no demands). Let $I = \{0\} \cup C \cup S$ be the set of all the nodes in the problem where 0 denotes the depot. Each path from i to j $\forall i, j \in I$ has a nonnegative travel length l_{ij} and the use of a stop node $j \in S$ is associated to a nonnegative cost f_j.

First, consider that the vehicle k starts the route leg at the depot or at a stop node that has been reached at an earlier stage. At the first visited node $j \in C \cup S$ on this leg, the cumulative length is $a_{jk} = l_{ij}$, where $i \in \{0\} \cup S$ is the initial node of the leg. For example, in Figure 3d for $j = 1, 3$, we have that $a_{11} = l_{01}$ and $a_{31} = l_{s3}$. Formally, these constraints are:

$$a_{jk} \leq L - (L - l_{ij})x_{ijk} \quad \forall i \in \{0\} \cup S, \forall j \in I, k = 1, 2, \ldots, K, \tag{1}$$

$$a_{jk} \geq l_{ij}x_{ijk} \quad \forall i \in \{0\} \cup S, \forall j \in I, k = 1, 2, \ldots, K, \tag{2}$$

where x_{ijk} is a binary variable that takes value one if arc (i, j) is used by vehicle k in the solution and value zero otherwise. Note that when $x_{ijk} = 1$, then $a_{jk} = l_{ij}$.

Now, for any path starting at a customer $i \in C$ along the route of vehicle k, the cumulative costs are computed as $a_{jk} = a_{ik} + l_{ij}$ if the arc from i to j is used by vehicle k. Formally,

$$a_{jk} \leq a_{ik} + L - (L - l_{ij})x_{ijk} \quad \forall i \in C, \forall j \in I, k = 1, 2, \ldots, K, \tag{3}$$

$$a_{jk} \geq a_{ik} - L + (l_{ij} + L)x_{ijk} \;\; \forall i \in C, \forall j \in I, k = 1, 2, \ldots, K. \tag{4}$$

Thus, if $x_{ijk} = 1$, then $a_{jk} = a_{ik} + l_{ij}$.

Finally, no cumulative cost can exceed the limit L:

$$a_{jk} \leq L \;\; \forall j \in \{0\} \cup S, k = 1, 2, \ldots, K. \tag{5}$$

An important remark must be made at this moment. These constraints are correct if the arc lengths l_{ij} satisfy the triangle inequality, that is, whether the costs are associated with Euclidean distances in $\mathcal{R}^2$. For the case of different distances (i.e, if the data do not verify the triangle inequality), we must adapt the constraints because a stop node could be visited because of being on the shortest path to reach a customer. In this case, there should be no cost for visiting the stop node. To solve this problem, copies of the stop nodes are introduced as customers without demands. Then, when the route visits a stop node, we have to consider if it is because the length of the leg is close to L and then the cost f_j must be computed, or it is because the shortest path for arriving at the next customer visits the stop node. In our paper, we will assume that the lengths l_{ij} satisfy the triangle inequality.

Note that these constraints may be added to any variant of vehicle routing problems. As it has already been exposed, we focus on problems with split delivery because of the real applications and savings in the cost solution with respect to the CVRP. Also, note that problems with at least one customer demand greater than the vehicle capacity cannot be solved as a CVRP but can solved as a SDVRP.

4. The Split Delivery Vehicle Routing Problem with Stop Nodes

Contrary to what is assumed in the CVRP, in the SDVRP each customer can be visited more than once and so its demand can be greater than the vehicle capacity. The remaining assumptions of the CVRP are still valid for the SDVRP.

First, we introduce some notations. Following with the sets C, S and I introduced in the previous section, the SDVRPSN can be described on a complete graph $G = (I, A)$, also called road graph, where A is the set of arcs $\{(i, j) |\, i, j \in I \text{ and } i \neq j\}$. A nonnegative cost c_{ij} and a nonnegative length l_{ij} are associated with each arc $(i, j) \in A$, representing the travel cost and the length of arc (i, j), respectively .

Each customer $i \in C$ is associated with a deterministic nonnegative demand d_i to be delivered (some of these demands might be greater than the vehicle capacity). The depot and the stop nodes have no demand. A set of K identical vehicles, each one with capacity $Q \in \mathbb{Z}^+$, is available at the depot. Each vehicle performs one route at most and we assume that K is equal to the minimum number of vehicles needed to serve all the customers K_{min}, which can be determined by the trivial lower bound of the CVRP, $K_{min} = \lceil d(C)/Q \rceil$, where $d(C)$ is the sum of the demands in C.

Next we present two mathematical formulations for the SDVRPSN: a vehicle flow formulation and a commodity flow formulation. The first one, also called formulation by arcs, uses the subtour elimination constraints defined on a subset of arcs to avoid cycles on the same route. The second one uses flow variables that define the incoming and outgoing flows of commodities at each node on the routes. Common notation for both formulations is the following:

- x_{ijk} is a binary variable for $i \neq j$ equal to one if vehicle k travels directly from i to j and zero otherwise, for $i, j \in I$;
- f_j is the cost of using a stop node for $j \in S$;
- L denotes the maximum route leg length, where $L \geq \max\{l_{0j}\}_{j \in A}$.

The objective function can be expressed as follows:

$$\min \sum_{i \in I} \sum_{j \in I} \sum_{k=1}^{K} c_{ij} x_{ijk} + \sum_{i \in I} \sum_{j \in S} \sum_{k=1}^{K} f_j x_{ijk}, \tag{6}$$

where the first term is the travel cost and the second term is the cost associated with the use of stop nodes in all the routes. Then, the optimal solution is the set of routes that minimizes the total serving cost.

4.1. Vehicle flow formulation

Defining the variable y_{ik} as the demand of customer $i \in C$ delivered by vehicle $k = 1, 2, \ldots, K$, the vehicle flow formulation for the SDVRPSN is:

$$\text{Min.} \quad \sum_{i\in I}\sum_{j\in I}\sum_{k=1}^{K} c_{ij}x_{ijk} + \sum_{i\in I}\sum_{j\in S}\sum_{k=1}^{K} f_j x_{ijk},$$

$$\text{s.t.} \quad \sum_{i\in I}\sum_{k=1}^{K} x_{ijk} \geq 1 \qquad \forall j \in C, \quad (7)$$

$$\sum_{j\in I}\sum_{k=1}^{K} x_{0jk} = K \quad (8)$$

$$\sum_{i\in I} x_{ipk} - \sum_{j\in I} x_{pjk} = 0 \qquad \forall p \in I, k = 1,2,\ldots,K, \quad (9)$$

$$\sum_{i\in D}\sum_{j\in D} x_{ijk} \leq |D| - 1, \qquad \forall D \subseteq C \cup S, D \cap S \neq \emptyset, k = 1,2,\ldots,K, \quad (10)$$

$$y_{ik} \leq d_i \sum_{j\in I} x_{ijk} \qquad \forall i \in C, k = 1,2,\ldots,K, \quad (11)$$

$$\sum_{i\in C} y_{ik} \leq Q \qquad k = 1,2,\ldots,K, \quad (12)$$

$$\sum_{k=1}^{K} y_{ik} = d_i \qquad \forall i \in C, \quad (13)$$

$$a_{jk} \leq L - (L - l_{ij})x_{ijk} \qquad \forall i \in \{0\} \cup S, \forall j \in I, \forall k = 1,2,\ldots,K, \quad (14)$$

$$a_{jk} \geq l_{ij}x_{ijk} \qquad \forall i \in \{0\} \cup S, \forall j \in I, \forall k = 1,2,\ldots,K, \quad (15)$$

$$a_{jk} \leq a_{iv} + L - (L - l_{ij})x_{ijk} \qquad \forall i \in C, \forall j \in I, \forall k = 1,2,\ldots,K, \quad (16)$$

$$a_{jk} \geq a_{ik} - L + (l_{ij} + L)x_{ijk} \qquad \forall i \in C, \forall j \in I, \forall k = 1,2,\ldots,K \quad (17)$$

$$x_{ijk} \in \{0,1\} \qquad \forall\ (i,j)\ \in A, k = 1,2,\ldots,K, \quad (18)$$

$$y_{ik} \geq 0 \qquad \forall i \in C, \forall k = 1,2,\ldots,K, \quad (19)$$

$$a_{jk} \leq L \qquad \forall j \in \{0\} \cup S, k = 1,2,\ldots,K. \quad (20)$$

Constraints (7) say that a customer must be visited by at least one vehicle while constraints (8) impose that exactly K vehicles leave the depot (if the model verifies the triangular inequality these constraints are not needed). Constraints (9) indicate that if vehicle k visits node p, then it must leave it and (10) are the classical subtour elimination constraints. Note that the subset D must have at least one stop node because the existence of subtours formed by only customer nodes are avoided by constraints (14)-(17). Constraints (11) guarantee that the quantity delivered by each vehicle does not exceed the demand of the node. Constraints (12) limit to Q the maximum load of each vehicle while constraints (13) ensure that the entire demand of each node is satisfied. Finally, constraints (14) to (17) and (20) are the route leg length constraints.

Note that y_{ik} must be a integer number, assuming that fractional services are not possible. In [6], the authors show that if the d_i are integer, then there always exists an optimal integer solution to the SDVRP, that is, variables y_{ik} can be relaxed from being integer to be

continuous.

A weakness of this formulation is the huge number of equations introduced by the constraints (10). A formulation that avoids this problem is the commodity flow formulation shown in next section.

4.2. Commodity flow formulation

Commodity flow models were first introduced by [21] for an oil delivery problem and then extended by [22] and [23] to variants of the Traveling Salesman Problem and the CVRP.

This formulation requires of new continuous variables associated with the arcs, which represent the amount of product that flow trough them. The commodity flow formulation is defined on an extended graph $G' = (I', A')$ obtained from G by adding a copy of the depot node as the vertex $n+p+1$.Thus, $I' = I \cup \{n+p+1\}$, $A' = \{(i,j), (0,i), (i,n+p+1) \,|\, i,j \in I' \backslash \{0, n+p+1\} \text{ and } i \neq j\}$. The arc costs associated to the copy of the depot are the same than the arc costs of the depot, that is $c_{i,n+p+1} = c_{0i}\ \forall i \in I' \backslash \{0, n+p+1\}$. Two non-negative flow variables w_{ijk} and w_{jik} are defined for each arc $(i,j) \in A'$. When vehicle k travels from i to j, w_{ijk} and w_{jik} represent the vehicle load and the residual capacity along arc (i,j) for vehicle k, respectively. Note that $w_{ijk} + w_{jik} = Q$ for every arc $(i,j) \in A'$ in a route k. Hence, the flow variables define two directed paths: one from the depot to its copy with the load of the vehicle and another in the contrary sense with its remaining capacity.

It is important to remark that the following formulation (a commodity flow formulation for the SDVRP) is new in the literature. In order to obtain it, the following constraints must be considered:

- Commodity Flow conservation constraints:

We know that the outgoing flow at node i is the incoming flow minus its demand. Thus, the difference between the incoming and the outgoing flows is d_i. Actually, since in a feasible solution a node $i \in C$ presents two incoming flows and two outgoing flows this difference is equal to $2d_i$. For $i \in S$ we have that $d_i = 0$ and then the condition also holds. Therefore, the constraints are:

$$\sum_{j \in I'} \sum_{k=1}^{K} w_{jik} - \sum_{j \in I'} \sum_{v=1}^{K} w_{ijk} = 2d_i \quad \forall i \in C, \tag{21}$$

and

$$\sum_{j\in I'}\sum_{k=1}^{K} w_{jik} - \sum_{j\in I'}\sum_{k=1}^{K} w_{ijk} = 0 \quad \forall i \in S. \tag{22}$$

• Outgoing flow from the depot:

Consider the flow path from the depot to its copy. By definition the outgoing flow w_{ijk} at node i represents the remaining demand to be delivered by vehicle k along the route until the end. Therefore, at the depot the outgoing flow w_{0jk} is the total demand delivered by vehicle k along the complete route. Summing over all the vehicles, this condition is:

$$\sum_{j\in I'\setminus\{0,n+p+1\}}\sum_{k=1}^{K} w_{0jk} = d(C). \tag{23}$$

• Incoming flow at the depot:

Consider the flow path from the copy of the depot to the depot. The incoming flow w_{jik} at node i is the remaining capacity of vehicle k after visiting the customer j. At the depot, this remaining capacity of the vehicle is the capacity Q minus the total demand delivered along the route by vehicle k. Summing over all the vehicles:

$$\sum_{j\in I'\setminus\{0,n+p+1\}}\sum_{k=1}^{K} w_{j0k} = KQ - d(C). \tag{24}$$

• Arc flow capacity: These constraints link the commodity flow variables and the vehicle flow variables:

$$w_{ijk} + w_{jik} = Q(x_{ijk} + x_{jik}) \quad \forall i, j \in I', k = 1, 2, \ldots, K. \tag{25}$$

• Outgoing flow from the copy of the depot:

The path defined from the copy of the depot to the depot represents the remaining capacity of the vehicle. Then the flow $w_{(n+p+1)jk}$ is the total capacity because no demand was delivered yet. For K vehicles we have that:

$$\sum_{j\in I'\setminus\{0,n+p+1\}}\sum_{k=1}^{K} w_{(n+p+1)jk} = KQ. \tag{26}$$

Thus, the full commodity flow formulation is:

$$\text{Min.} \quad \sum_{i\in I}\sum_{j\in I}\sum_{k=1}^{K} c_{ij}x_{ijk} + \sum_{i\in I}\sum_{j\in S}\sum_{k=1}^{K} f_j x_{ijk},$$

$$\text{s.t.} \quad \sum_{i\in I'}\sum_{k=1}^{K} x_{ipk} - \sum_{j\in I'}\sum_{k=1}^{K} x_{pjk} = 0 \qquad \forall p \in I\backslash\{0, n+p+1\}, \quad (27)$$

$$\sum_{j\in I'}\sum_{k=1}^{K}(w_{jik} - w_{ijk}) = 2d_i \qquad \forall i \in C, \quad (28)$$

$$\sum_{j\in I'}\sum_{k=1}^{K}(w_{ijk} - w_{jik}) = 0 \qquad \forall i \in S, \quad (29)$$

$$\sum_{j\in I'\backslash\{0,n+p+1\}}\sum_{k=1}^{K} w_{0jk} = d(C) \quad (30)$$

$$\sum_{j\in I'\backslash\{0,n+p+1\}}\sum_{k=1}^{K} w_{j0k} = KQ - d(C) \quad (31)$$

$$\sum_{j\in I'\backslash\{0,n+p+1\}}\sum_{k=1}^{K} w_{(n+p+1)jk} = KQ \quad (32)$$

$$(w_{ijk} + w_{jik}) = Q(x_{ijk} + x_{jik}) \qquad \forall i,j \in I, k = 1,2,\ldots,K \quad (33)$$

$$a_{jk} \le L - (L - l_{ij})x_{ijv} \qquad \forall i \in \{0\}\cup S, \forall j \in I'\backslash\{0\}, \forall k = 1,2,\ldots,K, \quad (34)$$

$$a_{jk} \ge l_{ij}x_{ijk} \qquad \forall i \in \{0\}\cup S, \forall j \in I'\backslash\{0\}, \forall k = 1,2,\ldots,K, \quad (35)$$

$$a_{jk} \le a_{ik} + L - (L - l_{ij})x_{ijk} \qquad \forall i \in C, \forall j \in I'\backslash\{0\}, \forall k = 1,2,\ldots,K, \quad (36)$$

$$a_{jk} \ge a_{ik} - L + (l_{ij} + L)x_{ijk} \qquad \forall i \in C, \forall j \in I'\backslash\{0\}, \forall k = 1,2,\ldots,K, \quad (37)$$

$$x_{ijk} \in \{0,1\} \qquad \forall\ i,j\ \in I', k = 1,2,\ldots,K, \quad (38)$$

$$y_{ik}, w_{ijk} \ge 0 \qquad \forall i \in C, j \in I', k = 1,2,\ldots,K, \quad (39)$$

$$a_{jk} \le L \qquad \forall j \in S\cup\{n+p+1\}, k = 1,2,\ldots,K. \quad (40)$$

The flow variables and the flow conservation constraints avoid the use of the subtour elimination constraints and, therefore, this formulation is smaller than the vehicle flow formulation. However, it gives weaker lower bounds.

5. Solution method

In relation to implementation, the exact solution of both models using general CPLEX's Branch and Bound requires a huge amount of time even for small examples (up to 7 customer nodes and 2 stop nodes using the instances of [32]). Hence, because of the complexity of the problem, we need an efficient algorithm to get a solution and to start the study of this problem. We present a tabu search heuristic with granular neighborhood. First, some

concepts about tabu search and granular searching are introduced.

5.1. Tabu search and granular local search

The neighborhood of a solution is a set of other solutions that can be obtained by simple modification of the current solution. This modification and the transition of a solution to a new one is called a *move*. In a simple local search, at each iteration, the best solution of the neighborhood that improves the current solution is introduced. Since after an improving move a new current solution is reached, the algorithm is iterated until no improving move is obtained. The current solution represents a local optimum in the current neighborhood. As it can be noted, the number of iterations needed to reach the local optimum may be large and depends mainly on the problem size. A weakness of a local search process is that, once the solution is close to a local optimum, it may be difficult to escape from it. The tabu search methodology deals with this problem.

The basic tabu search is based on procedures designed to cross boundaries of local optimality or feasibility guiding a local search procedure to explore the solution space beyond local optimality ([27]). The main component of the tabu search is its use of adaptive memory: to escape from a local optima and cycles, a tabu list or short term memory is used. This tabu list introduces the main attributes of a move and forbid in the current neighborhood all the moves that imply at least one attribute from the list. Moves at the tabu list are considered tabu, that is, they are forbidden during a certain number of moves. Intensification and diversification components are also important for a tabu search. During the intensification stage, the algorithm searches on a neighborhood of the elite solution by returning to attractive regions to search more thoroughly. On the diversification stage, the search is made on unvisited regions. After a given number of non-improving moves, the tabu list is stopped.

Although the tabu list is an effective tool for avoiding local optima, the time required to explore all possible neighborhoods may be very large, especially in large instances. For this reason, it is defined in [35] a restricted neighborhood, called granular neighborhood, obtained from the standard one by removing the moves that cannot belong to high quality feasible solutions. The granular neighborhood only looks for moves with an important probability of staying in the optimal solution at each iteration by defining elite neighborhoods. Because this variant examines a granular neighborhood in much less time than the original one, the time required to reach a highly quality solution is often much smaller. An important particularity of granular search is that the structure of the granular neighborhood may be modified during the evolution of the algorithm to diversify the search.

In [35], the authors observe that long arcs in the problem graph have a low probability of being part of high quality solutions. Therefore, a possible way to limit the search of the neighborhood is not considering moves that try to insert "long" arcs in the current solution. Following this idea, it is defined a new sparse graph that includes all the arcs that should be considered for insertion in the optimal solution. To define what "short" arc and a sparse graph are, a granular threshold value is defined as:

$$\upsilon = \beta \frac{z'}{(n+K)'}, \tag{41}$$

where β is a suitable positive sparsification parameter and z' is the value of any initial solution obtained by any fast algorithm. Note that the fractional term represents the average value of an arc in the initial solution. An arc (i, j) belongs to the sparse graph G' if:

$$c_{ij} \leq \beta \frac{z'}{(n+K)'}. \tag{42}$$

Finally, the sparse graph may also includes other arcs such as those from the depot to customers, which may exceed the granular threshold, defining the set Z. Then the sparse graph $G'' = (I', A'')$, where $A'' = \{(i, j) \in A' \,|\, c_{ij} \leq \upsilon\} \cup Z$ is defined.

By using the above ideas, we propose in this paper an algorithm called *Hybrid Granular Tabu Search Heuristic*. This algorithm can be used to solve SDVRP both with and without stop nodes.

5.2. The Hybrid granular tabu search algorithm

The details of the Hybrid Granular Tabu Search heuristic (HGTS) for the SDVRPSN are presented here. This algorithm has four main parts: 1) Initial Solution, 2) Granular Tabu Search, 3) Improving Solution, and 4) Route Leg Length correction. There is also a set of processes applied to improve the current solution based on local search and on the known properties of the problem. The algorithm is constructed for a symmetric Euclidean graph.

5.2.1. Initial solution

The initial solution is obtained with the well known Clarke and Wright save algorithm ([14]) based on the savings of merging two simples route $(0, i, 0)$ and $(0, j, 0)$ into one route $(0, i, j, 0)$. The saving for nodes $i, j \in C$ is defined as $s_{ij} = c_{0i} + c_{0j} - c_{ij}$. A

decreasing ordered list (or matrix) of these savings is built storing at each row the information of nodes i, j and the saving s_{ij}. Nodes are introduced and connected on the same route following the order in the list until the capacity of the vehicle is full. Once a node is added to a route, all the rows of the list that contain it are erased. If a vehicle cannot serve the full demand of a node, it is partially served by this route (until completing the vehicle capacity) and the route is over; the remaining demand will be served by other vehicle. Note that only the last nodes introduced in a route can be visited by more than one vehicle. In case of partial service, the rows on the list containing this node are not erased. Then the algorithm constructs4 K routes, where $K-1$ vehicles are full in capacity. If route K has a split node with a route z, the node is deleted from z and all its demand is allocated to the last vehicle (if the remaining capacity allows it). This initial solution is a feasible solution that uses the minimum number of vehicles. Before the construction of the routes with the Clarke and Wright algorithm, we perform single routes to nodes with demands greater than or equal to Q so that we reduce the set of nodes and the number of vehicles to generate the remaining routes. Finally, note that some of these routes can present route leg length greater than L.

These routes can be improved in the sense of having less cost by making moves: exchanges of nodes, splitting nodes or deleting nodes from some routes. The granular tabu search phase includes these procedures.

5.2.2. Granular tabu search phase

In this phase, seven different local search procedures are executed and a tabu list is built. Each procedure searches for an improving move in a granular neighborhood of the current solution and the best move, in sense of the savings, is introduced to obtain a new current solution. These moves are introduced at the tabu list and their inverse moves are tabu for the next t moves. Then, t represents the tabu list size and also we indicate with T the number of iterations with no moves to introduce in the solution. For the local search procedures, a granular neighborhood is defined to reduce the search space. The algorithm uses the following granular definition:

$$Gran = \beta \frac{z}{n+K+s}(1+\frac{RemCap_k}{AverRemCap}), \tag{43}$$

where z is the value of the current solution, s is the number of nodes with split in the current solution, $RemCap_k$ is the remaining capacity of vehicle k, we say $AverRemCap$ represents the average of the remaining capacity of routes with unused capacities. This new term into the $Gran$ definition allows to search in routes that are not in the normal granular space but

that are attractive because they have a relevant remaining capacity. Denoting with r_k the route performed by vehicle k, we say that a node $i \in r_k$ stays into the granular neighborhood of $j \in r_v$ if $c_{ij} \leq Gran$ and we denote it as $i \in G(j)$. Finally, y^0_{ik} and y^1_{ik} are the demands of customer i served by vehicle k before and after introducing a move, respectively.

The procedures used in our tabu search are the following:

- *Relocate node (Relocate)*: for customer nodes $i \in r_k$ and and $j \in r_v$, if either $i \in G(j)$ or $i \in G(j+1)$ and if the remaining capacity of r_v is large enough, then move the node i from r_k to r_v and locate it at the position $(j+1)^{th}$ on r_v. Demands are set as following: $y^1_{ik} = 0$ and $y^1_{iv} = y^0_{ik}$ (see Figure 4).

- *Exchange node (Exchange)*: for customer nodes $i, t \in r_k$ and $j, g \in r_v$, if either $i \in G(g)$ or $i \in G(g+1)$, either $j \in G(t)$ or $j \in G(t+1)$ and the remaining capacities of r_k and r_v are large enough, then move node i from r_k to r_v, locating it in the position $(g+1)^{th}$, and move j from r_v to r_k, locating it in position $(t+1)^{th}$. Demands are set as following: $y^1_{ik} = 0, y^1_{iv} = y^0_{ik}$ and $y^1_{jv} = 0, y^1_{jk} = y^0_{jv}$ (see Figure 5).

- *2-Opt (2opt)*: for customer nodes $i \in r_k, j \in r_v$, if $i \in G(j+1)$ and $j \in G(i+1)$ switch the initial part of $r_k(r_v)$ until node i (node j) and reconnect it with $j+1$ on r_v ($i+1$ on r_k). Demands are set as following: $y^1_{zv} = y^0_{zk}$ ($y^1_{zk} = y^0_{zv}$) where z takes the value of the first elements of r_k (r_v) up to node i (j). This move is done if the capacities of vehicles k and v allow it (see Figure 6).

- *Exchange Split (ExchSplit)*: for a customer node i visited by routes r_k and r_v (this is, $i \in r_k$ and $i \in r_v$) and customer nodes $z, z+1 \in r_v$, $j \in r_k$, if either $j \in G(z)$ or $j \in G(z+1)$, then erase i from r_v and generate a new split between these routes at a node j inserting it at position $(z+1)^{th}$ on r_v. Demands are set as following: $y^1_{iv} = 0, y^1_{ik} = y^0_{ik} + y^0_{iv}, y^1_{jk} = y^0_{jk} - y^0_{iv}$ and $y^1_{jv} = y^0_{iv}$ (see Figure 7).

- *Delete split and new split (DelSplitNew)*: for a customer node i visited by routes r_k and r_v and customer nodes $j \in r_k$ and $z, z+1 \in r_t$ if either $j \in G(z)$ or $j \in G(z+1)$ and the remaining capacity of r_t is large enough, erase i from r_v and generate a new split on node j between route k and a new route t, locating it at the position $(z+1)^{th}$ on r_t. Demands are set as following: $y^1_{iv} = 0, y^1_{ik} = y^0_{ik} + y^0_{iv}, y^1_{jk} = y^0_{jk} - y^0_{iv}$ and $y^1_{jt} = y^0_{iv}$ (see Figure 8).

- *Move splits (MoveSplit)*: given a customer node i that is visited by more than one vehicle, let B be the set of routes that have i in their paths and let B' be the set of routes without i in their paths. If for some $g \in r^*$, either $i \in G(g)$ or $i \in G(g+1)$ where r^* is a route in B' with the minimum insertion cost (the cost of introducing some

node on a route) of node i and the remaining vehicle capacity of r^* is large enough, then delete $i$from all route in B and introduce it in r^* at position $(g+1)^{th}$. Demands are set as following: $y^1_{ik} = 0, \forall k \in B$ and $y^1_{ir^*} = d_i$ (see Figure 9).

- *Delete best split (DelBestSplit)*: given a customer node i that is visited by more than one vehicle, let B be the set of routes that contain node i in their paths and let $r_k \in B$ be the route with the greatest saving cost for erasing i. If the total remaining capacities of $r_j, j \in B \backslash \{r_k\}$, are large enough, delete i from r_k and allocate the free demand y^0_{ik} to the other routes in B starting by the route with the lowest saving cost for erasing i and continuing with the next lowest saving cost and so on, until having allocated all y^0_{ik}. In other words, sort B by decreasing saving cost, delete i from the first route in B and split y^0_{ik} among the other routes in B starting by the last one and, when it is full, follow with the next one and so on (see Figure 10).

Figure 4: *Relocate* process.

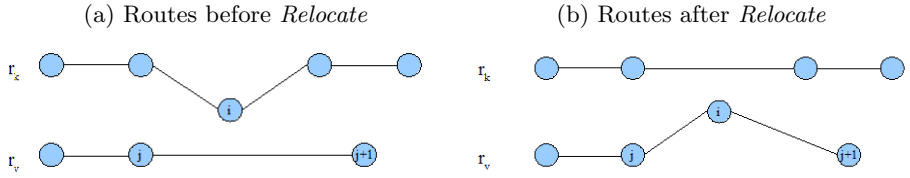


Figure 5: *Exchange* process.

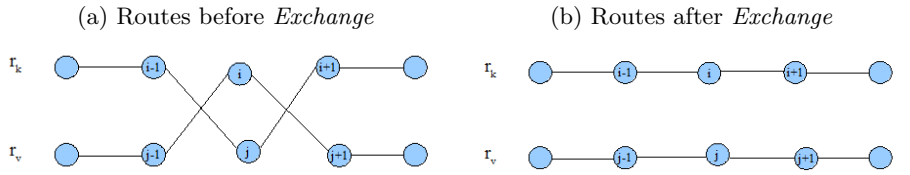
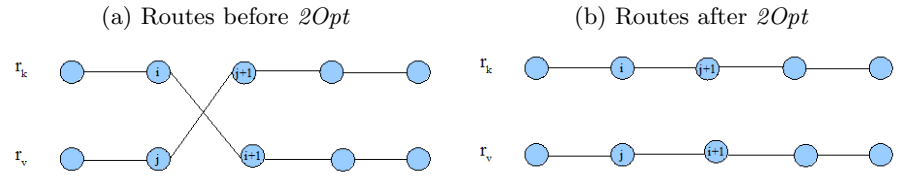


Figure 6: *2Opt* process.



The procedures *Relocate, Exchange* and *ExchSplit* are also applied in [28]. *2-Opt* is the classical procedure developed in [15] for the traveling salesman problem. *DelBestSplit* is presented in [8], but they use the insertion criteria in the inverse sense: starting by the route with the greater saving cost. We use a different criterion because a route with a lower saving

Figure 7: *ExchSplit* process.

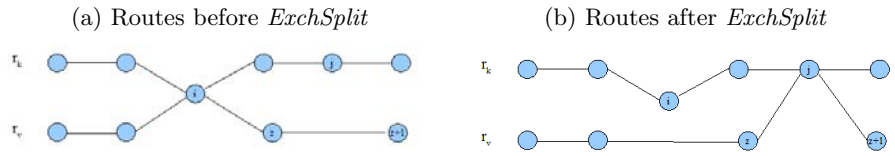


Figure 8: *DelSplitNew* process.

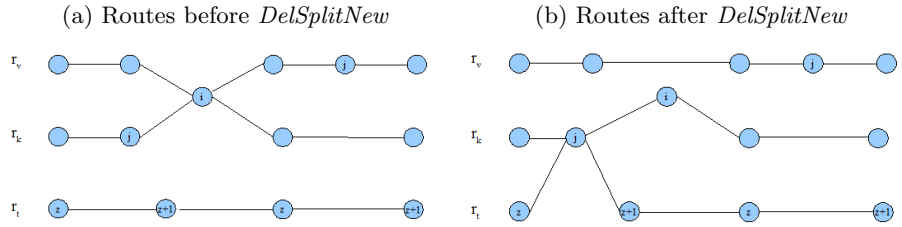


Figure 9: *MoveSplit* process.

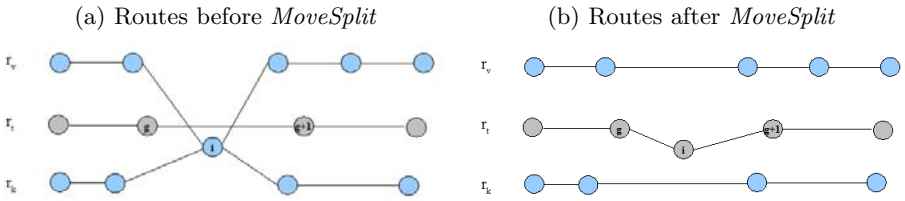
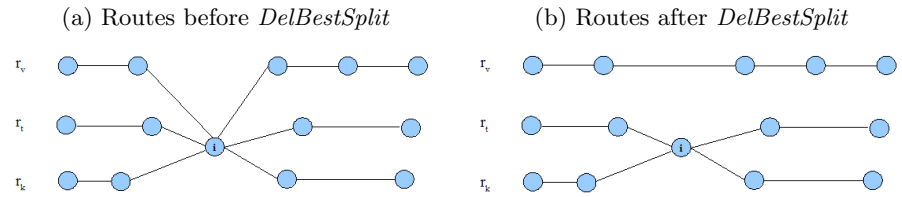


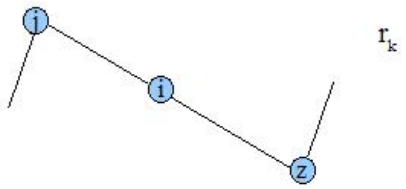
Figure 10: *DelBestSplit* process.



cost for erasing i has a small chance to delete i in some other move. Therefore we can say that node i is well located in this route. Think of a route k with node i located in the direct path between nodes j and z (see Figure 11). In this case, the saving cost for erasing i is equal to zero, the lowest saving cost possible because the triangular inequality.

Each procedure uses the same current solution and calculates the saving cost of the potential moves if they are not in the tabu list. They are sorted in a list and the algorithm selects randomly one among the best m candidate moves by assigning higher probabilities to the best moves. This means that if a certain procedure has more than one possible move,

Figure 11: Route with lowest saving cost of erase i



then the selected one might not be the best one from the list. The goal of this strategy is to escape from local optima reached if choosing always the best move. Then, once the algorithm has the "proposed" moves of each procedure, the best one in sense of saving cost is introduced in the solution. The inserted move is stored in the tabu list and the reverse move is forbidden for t moves. After $T-1$ iterations without moves with positive saving costs, the algorithm allows to introduce a nonpositive move if there is at least one. Because of that we call the algorithm *hybrid*: the general tabu search algorithm allows nonpositive moves (with negative saving cost) at every iteration. Finally, if no move has been added after T iteration the tabu phase is over. For intensification, if an improving move is found at next iteration, the value of β in *Gran* is reduced in $\phi\%$ to intensify the search of this neighborhood. Also for diversification, if no improving move is found, the value of β is increased in $\phi\%$ in the next iteration to explore a new neighborhood.

The saving cost for each move is weighted by a factor that represents the remaining capacity in the route where the node will be inserted. For example, in *Relocate* the saving cost is weighted by the remaining capacity of R_v after having introduced the node i in the path. Thus, the saving cost at each procedure is

$$sc_p = (\sum Cost_{delarcs} - \sum Cost_{addarcs})(RemCap_v + dem_{ik}), \tag{44}$$

where the parenthesis contains the difference between the sum of deleted arcs and added arcs due to the move and dem_{ik} represents the demands exchanged from route k to v.

Other important idea is that the algorithm can explore neighborhoods of infeasible solutions in the sense of both vehicle capacity and route leg length. The infeasibility in capacity is controlled by a parameter π such that the total demand served on a route cannot exceed the threshold $\pi Q, \pi \geq 1$. Then, the tabu phase relaxes constraints (12) of the vehicle flow formulation to

$$\sum_{i \in C} y_{ik} \leq \pi Q, \quad \forall k = 1, \ldots, K. \tag{45}$$

This infeasibility will be fixed in the next phase of the algorithm called Capacity Correction phase.

5.2.3. Capacity correction

This phase is active if, and only if, certain route k serves demands exceeding the capacity Q. This process presents a set of moves from route k to one, two or three different routes trying to eliminate the excess of capacity. It combines different ways to make new splits, relocations or splits and relocations using up to three different nodes from route k. If there is more than one possible move, the best in term of saving cost is introduced. The phase is over when there is no route with excess of capacity. Note that the solution can still be infeasible with regard to the route leg length.

5.2.4. Improving solution process and granular local search improving process

After the capacity correction, the algorithm improves the given solution with three procedures: *2-splits cycle, Node-position and Subtours-elimination.* The *2-splits cycle* procedure eliminates solutions where two routes share two split nodes. *Node-position* relocates nodes of a route in a different position if a saving is possible. *Subtour-elimination* procedure deletes subtours in a given solution by erasing multiple visits of a vehicle to the same customer.

Finally, the algorithm applies a granular local search with the same procedures of the tabu phase but setting π to one. Then, only infeasible solutions in term of route leg length are allowed.

5.2.5. Route leg length correction

The solution obtained might be infeasible in terms of route leg length, that is, some route leg length can exceed L. Because the algorithm assumes symmetric data (i.e, $c_{ij} = c_{ji}$), it explores the route with excess length in both orientations, getting two candidates solutions for each route k, r_{k_1} and r_{k_2}. It introduces the "best" stop nodes (in the sense of insertion cost) before the node where the route leg exceeds L and checks that a_{sk} and a_{0k} are less

than or equal to L. When the needed stop nodes are introduced and there are no route leg lengths longer than L, the algorithm compares r_{k_1} and r_{k_2} and selects the shortest one.

The full algorithm works as follows:

1. Initial Solution: Clarke and Wright algorithm: *CurrentSolution*
2. Repeat M times:
   - 2.1 Clear the Tabu-list.
   - 2.2 While ($NoMove \leq$ T):
     - 2.2.1 GRANULAR TABU: Construct the Tabu-list.
       - 2.2.1.1 Select randomly one from m best moves of:
         - *Relocate.*
         - *Exchange.*
         - *2opt.*
         - *DelSplitNew*
         - *ExchSplit*
         - *MoveSplit*
         - *DelBestSplit*
       - 2.2.1.2 if NoMove<T-1 only positive moves are allowed
       - 2.2.1.3 Update *CurrentSolution*
     - 2.2.2 If there is at least one move: NoMove=0 and $\beta = (1 - \phi/100) * \beta$.
       If there is no move: NoMove=NoMove+1 and $\beta = (1 + \phi/100) * \beta$.
     - 2.2.3 If NoMove= T-1, allow negative moves.
   - 2.3 Improving Solution processes: *2-cycles, Subtours, Node-position.*: update *CurrentSolution.*
   - 2.4 If $R_k, k = 1, 2 \ldots, K$ exceeds Q, *Capacity correction process*: update *CurrentSolution.*
   - 2.5 Improving Solution processes: *2-cycles, Subtours, Node-position.*: update *CurrentSolution.*
3. Local search improve process: update *CurrentSolution.*
4. Route leg length correction: ***BestSolution***

Note that the algorithm presents an important number of local search procedures that may demand expensive computational time but we balance this weakness by exploring

through granular neighborhoods. Also the correction capacity phase includes different moves to find a good correction for the infeasibility on vehicle capacity. Finally, note that the algorithm introduces the route leg length in the last phase and therefore, we can solve the SDVRP using the HGTS.

6. Computational results

First, we want to evaluate the performance of HGTS with the SDVRP. Because there are no benchmarks for the SDVRPSN, we solve this routing problem without constraints on the route leg lengths, that is the SDVRP. This is equivalent to use a very large value of L such that the stop nodes are not necessary. In this case, the solutions obtained can be comparable with the state-of-the-art heuristics for solving the SDVRP. Then, the value of L is included and we study the SDVRPSN. We use the same set of instances than [8], available in the web page of the authors (http://www.unibs.it/on-line/dmq/Home/Personale/articolo2398.html).

6.1. About the instances and benchmarks

In [8], problems 1-5, 11, and 12 from [26] are considered. The number of customers varies from 50 to 199 and the vehicle capacity from 140 to 200. Five additional sets of instances are created by changing the demands of the customers in the basic instances, but keeping all the other characteristics. Each of the new sets of instances is characterized by a lower bound and upper bound on the customer demand, α and γ respectively, expressed as a fraction of the vehicle capacity Q and $\alpha \leq \gamma$. Thus, the demand d_i of customer i is:

$$d_i = \alpha Q + \delta(\gamma - \alpha)Q, \tag{46}$$

for some random value $\delta \in [0, 1]$. Therefore, the demand d_i of customer i is set randomly in the interval $[\alpha Q, \gamma Q]$. The following lower and upper bound combinations are used to construct the demands of these new instances (α, γ)= $(0.01, 0.1)$, $(0.1, 0.3)$, $(0.1, 0.5)$, $(0.1, 0.9)$, $(0.3, 0.7)$ and $(0.7, 0.9)$.

The solution benchmarks for these instances are the results found in [2], [11] and [12]. These papers use the original instances of [26] and generate the random demands with the methodology described in [8]. Therefore, we cannot make an accurate comparison with them. For this reason, we compare the improving percentage of these algorithms with respect to the tabu search of [8] with the improving precentages of the results of HGTS with respect

the results published in [8]. Additional, by since the results of [8] are improved in [5], with the same set of instances, we also compare our results with this paper. To generate the stop node data, the space where the nodes are located is divided into a grid of different size depending on the number of customer nodes. In each cell of the grid we generate a random number that represents the stop node location.

6.2. Results

Our algorithm presents 7 parameters to be set: 1) the tolerance on the excess of capacity for infeasible solutions (π), 2) the tabu list size (t), 3) times that the tabu process is repeated (M), 4) candidate list (m), 5) number of iterations with no moves to add (T), 6) the granular coefficient (β) and 7) the intensification/diversification parameter (ϕ). After testing different parameter combinations, taking into account the trade-off between the solution quality and the CPU-time, we select the following values: $\pi = 1.10$ for problems P1 and P2 (those with lower demands with respect to the vehicle capacity) and $\pi = 1.05$ for P3-P5 and P11, $t = 5$, $M = 30$ for P5 and 50 for the remaining problems, $m = 5$, $T = 5$, $\beta = 2$ and $\phi = 10\%$.

6.2.1. SDVRP results

Table 1 shows the main results comparing the solution of HGTS with the Tabu search (TABUSPLIT) of [8] and the Optimization based heuristic (OpBH) of [5] for the SDVRP. The first two columns contain information about the demand parameters α and γ, the problem name and the number of customers; columns 3 and 4 are the values obtained by the TABUSPLIT with respect to the number of vehicles and objective value (mean value over 5 runs), respectively (obtained from the already mentioned webpage of the authors). Columns 5, 6 and 7 give the solution of OpBH found in the original paper: column 5 contains the number of vehicles, column 6 the mean value of the objective and column 7 the best solution (using the best set of parameters for each instance). Columns 8, 9 and 10 are our results with HGTS: column 8 is the number of vehicles used at each instance (always the minimum), column 9 is the mean of the objective value over 5 runs and column 10 is the best solution from these 5 runs. Finally column 11 presents the improving percentage with respect to OpBH, except for problems with demand parameters (0.01,0.1) because they are not reported in [5]. For theses instances we compare with TABUSPLIT. The problems are grouped by demand parameters.

The improving percentage μ is calculated as:

$$\mu = (\frac{Obj_{HGTS}}{Obj_{OpBH}} - 1) * 100, \tag{47}$$

where Obj_{HGTS} and Obj_{OpBH} are the solutions of the HGTS and OpBH respectively. Negative values of μ means an improvement at the solution.

We refer to a specific instance of problem Px with demand parameters (α, γ) as $Px(\alpha, \gamma)$. HGTS solves better than OpBH problems marked with bold letters at column 11. For instances with the original demands, HGTS improves solution of problems P1 and P2, the ones with lower number of customers. Instances with demand parameters (0.01,0.1) are not better using the HGTS. The remaining instance results present improvements in different ways except for problems P1(0.1,0.3), P1(0.1,0.9), P2(0.1,0.3) and P3(0.1,0.5). Results marked with asterisks mean that we reduce the number of vehicles used. Therefore, for problems with demand parameters between (0.1,0.3) and (0.7,0.9), HGTS gets better solutions in objective value in 16 instances (bold letter), in number of vehicles in 3 instances (asterisks) and in both objective value and vehicles in 8 instances (bolt letter and asterisk). Note that in some instances the reduction of the number of vehicles is important: in P11(0.1,0.9) and P5(0.3,0.7) two vehicles are saved and in P11(0.7,0.9) the fleet is reduced in three units.

As a disadvantage of HGTS, we can mention that it does not use the GENIUS algorithm developed in [25] for improving the solution of the traveling salesman problem, which is a very efficient insertion procedure and a postoptimization routine. In [8] the authors implement it to improve their solution. From our point of view, adding this procedure might help to improve the solution but computational time and complexity will increase. CPU times for HGTS are lower than OpBH for all instances but we do not mention it as an advantage because the differences in technical characteristics on the computers used. Detailed information on CPU times for HGTS can be found in Table 3.

As it was mentioned above, papers [11], [12] and [2] use this set of problems but they generate their own demands. To evaluate the performance of the HGTSm, we compare the used vehicles and the improving percentage with respect to TABUSPLIT solution. Table 2 reports the best known solutions for these instances found in [11] with a Memetic Algorithm (MA, in columns 3 and 4), [12] with Scatter Search (SS, in columns 5 and 6) and [2] with Tabu Search with Vocabulary Building Approach (TSVBA, in columns 7 and 8). Columns 9 and 10 of Table 2 are the results of HGTS. Now, the improving percentage μ is calculated as:

$$\mu = (\frac{Obj_{alg_X}}{Obj_{TABUSPLIT}} - 1) * 100, \tag{48}$$

Table 1: Solution of the SDVRP with different algorithms

Instances		TABUSPLIT		OpBH			HGTS			HGTS vs OpBH
α: 0	γ: 0	K	z	K	z	z min	K	z	z min	μ (%)
P1	50	5	5307907	5	5276590	5276751	5	5393358	5269262	**-0,14**
P2	75	10	8542757	10	8535923	8536078	10	8468316	8391745	**-1,69**
P3	100	8	8413577	8	8412736	8401150	8	8624538	8522229	1,44
P4	150	12	10708613	12	10635794	10550759	12	10868161	10700740	1,42
P5	199	16	13403505	16	13383400	13383599	16	13893905	13727158	2,57
P-11	120	7	10569587	7	10569587	10569587	7	10934244	10730883	1,53
α: 0.01	γ: 0.10									
P1	50		4629056	-	-	-	3	4810481	4641271	0,26
P2	75		6239394	-	-	-	5	6670397	6484007	3,92
P3	100		7714649	-	-	-	6	8062067	7926288	2,74
P4	150		9471386	-	-	-	9	9882560	9606843	1,43
P5	199		11482700	-	-	-	12	12018282	11918258	3,79
P-11	120		10552825	-	-	-	8	11402564	11231752	6,43
α: 0.1	γ: 0.3									
P1	50	11	7653121	11	7653121	7582003	11	7737566	7729536	1,95
P2	75	16	11340760	16	11228487	11229145	16	11374473	11323738	0,84
P3	100	22	15151732	22	15069913	15054586	22	15013107	14852841	**-1,34**
P4	150	32	21018042	32	20967599	20932806	32	20745296	20605262	**-1,56**
P5	199	41	25858494	41	25832635	25826172	41	25571429	25486903	**-1,31**
P-11	120	26	30604668	26	30350649	30179211	26	29557770	29347110	**-2,76**
α: 0.1	γ: 0.5									
P1	50	16	10391059	16	10373394	10210207	16	10287865	10219025	0,09
P2	75	24	15566936	24	15492215	15485438	24	15197313	15150386	**-2,16**
P3	100	33	20541296	33	20251664	20245791	33	20374753	20325598	0,39
P4	150	49	29916416	49	29817692	29770034	49	29319452	29131089	**-2,15**
P5	199	63	36242004	63	36064418	35939960	63	35553718	35415731	**-1,46**
P-11	120	40	45026152	40	44940602	44763774	40	42956594	42652382	**-4,72**
α: 0.1	γ: 0.9									
P1	50	26	15119826	26	15119826	14972843	26	15117659	15065504	0,62
P2	75	41	23386654	41	23379638	23378115	40	23560322	23382550	0,02*
P3	100	56	31552228	56	31413398	31362933	56	31425076	31249365	**-0,36**
P4	150	84	46741320	84	46624467	46599000	83	46268072	45889819	**-1,52***
P5	199	107	57158484	107	57129905	57102065	105	56438751	56299422	**-1,41***
P-11	120	67	73501136	69	73392354	71172434	67	69569315	69415274	**-2,47***
α: 0.3	γ: 0.7									
P1	50	26	15039466	26	15020022	15020022	26	14988832	14911964	**-0,72**
P2	75	39	22935488	39	22905672	22631233	39	22627513	22458874	**-0,76**
P3	100	53	30709048	53	30555503	30555132	53	30361472	30113456	**-1,45**
P4	150	80	44968584	79	44662798	44654674	79	44090876	44050262	**-1,35**
P5	199	103	55711292	104	55538587	55497672	102	55860044	55677763	0,32*
P-11	120	65	71682608	65	71266849	71268363	64	67713770	67262705	**-5,62***
α: 0.7	γ: 0.9									
P1	50	42	21736308	42	21667970	21667970	41	21718227	21617517	**-0,23***
P2	75	61	32853678	62	32741975	32503861	61	32704207	32608448	0,32*
P3	100	82	44707136	82	44577485	44525487	82	44244666	44128578	**-0,89**
P4	150	123	64821904	123	64627438	64627754	122	64725917	64618811	**-0,01***
P5	199	162	83921136	162	83551883	83554528	161	83007324	82817770	**-0,88***
P-11	120	99	106733056	101	105484279	104297549	98	102918853	102676746	**-1,55***

where Obj_{alg_X} is the solution of the algorithm MA, SS, TSVBA or HGTS. Negative values of μ means an improvement at the solution of algorithm X and $Obj_{TABUSPLIT}$ is the objective value of the TABUSPLIT algorithm. We use the TABUSPLIT solutions available on the web-page of the authors for the comparison.

Problem set P11 is found in [11], [12] and [2] as P6. We do not report the solution of P12 (or P7) because it is not reported in [8]. Previous results show that MA seems to be the most efficient algorithm for solving the SDVRP because they improve almost all the solutions of TABUSPLIT, especially for demands lower than $Q/2$. SS reaches better improving percentages on the instances with original demands P5, P11 and P2(0.01,0.1), P3(0.01,0.1), P5(0.3,0,7) and P11(0.3,0.7). TSVBA also is a very good algorithm. Note that MA and TSVBA do not use the minimum number of vehicles but SS and HGTS do it. Because of that we present two comparisons: column 11 presents the algorithm that gets the best improving percentage among all algorithms (based on columns 4, 6, 8 and 10) and column 12 presents the best improving percentage using the minimum number of vehicles

(based on columns 6 and 10).

Bolt letters in column 10 mark that our algorithm gets the best improving percentage over all algorithms and asterisks mark when our algorithm finds the best solution using the minimum number of vehicles. This information is summarized in columns 11 and 12. Thus, for problems with demand parameters (0.1,0.5) HGTS gets the best improving percentages in problem P2 with an improving percentage of 2.68%. For problems with demand parameters greater than or equal to (0.1,0.9), HGTS reaches the best improving percentages in almost all the instances, except for P2(0.1,0.9) and P1, where MA presents the best improving percentage and P5(0.3,0.7), where SS has the best result. The reason why HGTS has these best results might be because this algorithm focuses on the split procedures and capacity corrections introducing many different moves into them. Note that when the customer demands are closer to Q, more moves related to splits and capacity correction must be used.

HGTS also reaches the best improving percentage using the minimum number of vehicles in all the instances with parameter demands greater than or equal to 0.10, except in P1(0.1,0.3), P2(0.1,0.3) and P5(0.3,0.7). Therefore, from among 42 instances, HGTS obtains the best improving percentage in 16 instances among all the algorithms and in 27 instances using the minimum number of vehicles.

Concerning the CPU time, the comparison might not be fair because of the important differences in computer characteristics used to run the algorithms. In [11], MA was executed on a 3 GHz PC; in [12], SS uses a 2.40 GHz PC with 1GB RAM and TVBA is run in a 2.84 GHz PC with 512 MB RAM in [2]. HGTS is executed on a 3 GHz PC with 4 GB RAM. Table 3 presents the CPU times for each approach.

6.2.2. SDVRPSN results

HGTS performs very well for the SDVRP. Next step is to reduce the value of L such that the set of stop nodes are necessary to get a solution. We use from all the problems, only those instances with the original demands and with demand parameters (0.01,0.1) and (0.1,0.3), because for the other instances the routes are "short" due to the relation of demands and vehicle capacity. The maximum route leg length is set to $L = 2 * \max\{c_{0i}\}_{i \in C}$ so that a single route to the farthest node is allowed.

Table 4 presents the SDVRPSN solutions. Columns 1 and 2 are the instance names and the demand parameters. Column 3 contains the total number of stop nodes in the data. The number of stop nodes used in the solution is presented in column 4 and column 5 presents

Table 2: Improving percentage respect to TABUSPLIT

Instances		MA		SS		TSVBA		HGTS			
		K	Improve (%)	K	Improve (%)	K	Improve (%)	K	μ (%)	Best	Best k min
α: 0	γ: 0										
P1	50	5	-1,68	5	-0,58	5	0,00	5	-0,73	MA	SS
P2	75	11	-3,02	10	-1,36	11	1,52	10	-1,77	MA	SS
P3	100	8	-0,74	8	-0,56	8	1,21	8	1,29	MA	SS
P4	150	12	-2,57	12	-1,12	12	2,13	12	-0,07	MA	SS
P5	199	17	-2,33	16	-2,40	17	-1,31	16	2,41	SS	SS
P-11	120	7	-1,4	7	-3,18	7	-2,33	7	1,53	SS	SS
α: 0.01	γ 0.1										
P1	50	3	-0,64	3	0,00	3	1,29	3	0,26	MA	SS
P2	75	4	-0,85	4	-1,18	4	1,65	5	3,92	SS	SS
P3	100	5	-3,38	5	-4,02	5	-2,06	6	2,74	MA	SS
P4	150	8	-1,72	8	-1,71	8	0,53	9	1,43	MA	SS
P5	199	10	-3,56	10	-2,12	10	2,29	12	3,79	MA	SS
P-11	120	6	-9,97	6	-8,23	6	-6,82	8	6,43	MA	SS
α: 0.1	γ 0.3										
P1	50	10	-1,31	10	-1,29	10	0,44	11	1,00	MA	SS
P2	75	15	-1,9	15	-1,07	15	0,20	16	-0,15	MA	SS
P3	100	20	-2,24	20	-1,51	20	-0,18	22	-1,97*	MA	HGTS
P4	150	30	-2,06	29	0,45	30	0,07	32	-1,96*	MA	HGTS
P5	199	39	-1,85	38	0,88	39	-0,16	41	-1,44*	MA	HGTS
P-11	120	24	-6,8	23	-3,84	24	-3,76	26	-4,11*	MA	HGTS
α: 0.1	γ 0.5										
P1	50	15	-2,02	15	0,38	15	2,94	16	-1,66*	MA	HGTS
P2	75	23	-2,07	22	1,58	23	1,24	24	**-2,68***	HGTS	HGTS
P3	100	29	-2,61	29	1,55	29	0,42	33	-1,05*	MA	HGTS
P4	150	45	-2,7	43	1,27	45	1,16	49	-2,63*	MA	HGTS
P5	199	56	-2,84	56	0,47	56	0,61	63	-2,28*	MA	HGTS
P-11	120	35	-6,46	34	-3,48	35	-2,73	40	-5,27*	MA	HGTS
α: 0.1	γ 0.9										
P1	50	26	-0,19	25	4,91	26	1,50	26	**-0,36***	HGTS	HGTS
P2	75	39	-1,03	37	3,27	39	2,44	40	-0,02*	MA	HGTS
P3	100	48	-0,47	48	2,94	48	2,25	56	**-0,96***	HGTS	HGTS
P4	150	74	3,48	71	2,12	74	4,36	83	**-1,82***	HGTS	HGTS
P5	199	93	1,8	93	1,37	93	1,21	105	**-1,50***	HGTS	HGTS
P-11	120	56	-4,03	56	-2,30	56	-3,05	67	**-5,56***	HGTS	HGTS
α: 0.3	γ 0.7										
P1	50	26	-1,33	25	2,55	26	0,75	26	-0,85*	MA	HGTS
P2	75	38	-1,31	37	2,87	38	0,73	39	**-2,08***	HGTS	HGTS
P3	100	49	-0,41	49	3,13	49	1,16	53	**-1,94***	HGTS	HGTS
P4	150	74	0,15	73	1,53	74	0,71	79	**-2,04***	HGTS	HGTS
P5	199	96	1,03	96	-1,27	96	-0,96	102	-0,06	SS	SS
P-11	120	58	-3,24	58	-3,57	58	-3,12	64	**-6,17***	HGTS	HGTS
α: 0.7	γ 0.9										
P1	50	41	-0,5	40	6,40	41	2,12	41	**-0,55***	HGTS	HGTS
P2	75	60	0,62	60	5,26	60	1,86	61	**-0,75***	HGTS	HGTS
P3	100	80	0,25	80	4,91	80	2,38	82	**-1,29***	HGTS	HGTS
P4	150	119	1,15	119	2,29	119	1,29	122	**-0,31***	HGTS	HGTS
P5	199	158	1,72	158	1,42	158	1,54	161	**-1,31***	HGTS	HGTS
P-11	120	95	-2,34	95	-1,44	95	-1,17	98	**-3,80***	HGTS	HGTS

the number of stop nodes in the routes that use at least one stop node. Columns 6 and 7 present the mean objective value over 5 runs and the minimum value of those, respectively. The objective value includes the cost of using the stop nodes, set as 8000 for every stop node. Finally, column 8 shows the increase of the cost for the new constraints with respect to the best solutions of the HGTS for the SDVRP in Table 1. Note that P3, P4 and P5 with demands with parameters (0.10,0.30) do not use the stop nodes. All the tested instances use more than one stop node, and also, some of them present routes that use more than one stop node. The total cost increases between 0.69% and 9.74% respect with the best solution of the SDVRP.

Figure 12 shows the solution for P1(0,0). Circles points denote customers, square points are the stop nodes and those in black color are the stop nodes used for the vehicles. Note that the used stop nodes are those that minimize the distance between two nodes on a route, such that the route leg length constraint is not broken.

Table 3: CPU time (s)

		MA	**SS**	**TSVBA**	**HGTS**
α: 0	γ: 0				
P1	50	8,53	49,7	49,84	29,25
P2	75	35,72	166,5	145,78	51,75
P3	100	34,59	276,1	295,22	71,44
P4	150	103,69	527,1	2217,17	276,87
P5	199	353,84	588,3	4514,28	110,2
P-11	120	50,92	270,3	1944,19	654,5
$\alpha : 0.01$	$\gamma : 0.1$				
P1	50	12,38	51,8	19,69	45,75
P2	75	18,75	144	134,14	181,5
P3	100	37,12	272,1	1944,19	158,25
P4	150	100,27	743,3	2640,95	351,99
P5	199	356,22	1874,8	11215,52	872,17
P-11	120	72,98	370,9	2736,34	1054,21
$\alpha : 0.1$	$\gamma : 0.3$				
P1	50	10,22	66,4	23,17	9
P2	75	34,14	143,8	97,17	12,9
P3	100	78,06	305,1	160,95	27,13
P4	150	147,86	326,6	755,08	75,42
P5	199	347,14	32,1	1544,36	174,54
P-11	120	144,19	380,8	463,97	53,94
$\alpha : 0.1$	$\gamma : 0.5$				
P1	50	12,49	87,1	17,72	30,75
P2	75	37,38	126,8	67,66	14,24
P3	100	28,39	225,2	145,05	43,98
P4	150	224,89	21,3	470,34	83,26
P5	199	436,2	31,2	1216,69	238,34
P-11	120	163,14	329	340,53	54,4
$\alpha : 0.1$	$\gamma : 0.9$				
P1	50	21,42	92,6	19,11	39
P2	75	46,11	119,9	61,81	29,5
P3	100	84,38	177,9	125,28	89,35
P4	150	244,91	50,4	451,95	274,41
P5	199	725,69	50,7	108,63	534,51
P-11	120	196,14	20,6	418,98	151,56
$\alpha : 0.3$	$\gamma : 0.7$				
P1	50	24,853	92,4	19,09	29,25
P2	75	51,78	11,1	55,17	39,5
P3	100	100,16	17	134,84	84,71
P4	150	244,86	23	449,34	219,65
P5	199	749,94	327,3	119,04	40,25
P-11	120	271,39	20,5	436,8	108,98
$\alpha : 0.7$	$\gamma : 0.9$				
P1	50	22,91	5,8	24,41	31,5
P2	75	27,48	10,5	86,27	66,4
P3	100	55,75	38,3	185,55	86,98
P4	150	401,62	30,5	678,94	355
P5	199	571,7	215	153,12	274,25
P-11	120	298,08	20,4	30,32	195,06

Table 4: HGTS for SDVRP with Stop nodes

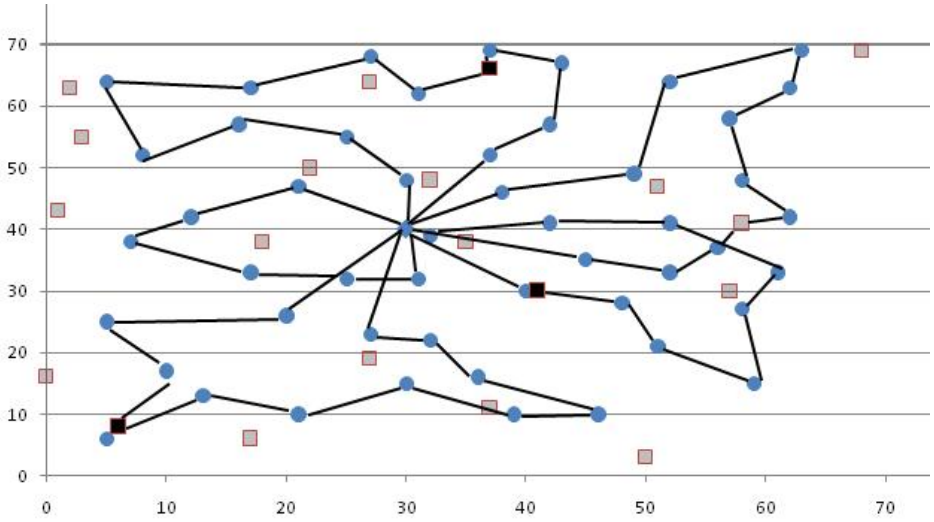
α: 0	γ: 0	SN	SN used	Routes	z	z min	Cost (%)
P1	50	20	3	$R_1, R_2, R_3 : \{1\}$	5588912	5419797	2.86
P2	75	20	4	$R_1, R_2, R_3, R_4 : \{1\}$	9593272	9208852	9.74
P3	100	30	4	$R_2, R_3, R_4, R_6 : \{1\}$	8850630	8782165	3.05
P4	150	56	6	$R_1, R_2, R_3, R_6, R_{11} : \{1\}$	11250866	10970295	2.52
P-11	120	36	3	$R_2, R_3, R_4 : \{1\}$	10944934	10853018	1.14
$\alpha : 0.01$	$\gamma : 0.1$						
P1	50	20	4	$R_3 : \{1\}, R_1, R_2 : \{2\}$	5078419	5067974	9.18
P2	75	20	5	$R_1, R_3, R_5 : \{1\}, R_2 : \{2\}$	6659760	6568705	1.31
P3	100	30	6	$R_1, R_2, R_3, R_5 : \{1\}; R_4 : \{2\}$	8319168	8240947	3.97
P4	150	56	6	$R_1, R_2, R_4, R_5, R_7, R_8 : \{1\}$	10114248	9901622	3.07
P-11	120	36	3	$R_1, R_2, R_3 : \{1\}$	11384465	11308892	0.69
$\alpha : 0.1$	$\gamma : 0.3$						
P1	50	20	3	$R_1, R_2, R_3 : \{1\}$	7934106	7893430	2.12
P2	75	20	3	$R_6, R_7, R_13 : \{1\}$	12498145	12038303	6.31
P3	100	30	-	-	-	-	-
P4	150	56	-	-	-	-	-
P-11	120	36	-	-	-	-	-

7. Conclusions and further research

In this work, it has been introduced a new variant of the vehicle routing problem where a route leg length cannot exceed a given length L. Thus, a new set of nodes with no demands, called stop nodes are defined. The vehicles can visit a stop node to avoid the breaking of this condition. A new set of constraints are defined and added in the SDVRP, model taken as the base of our study because it can generate savings with respect to the model with no splits.

Two mathematical formulation have been derived (a vehicle flow formulation and a commodity flow formulation) and a heuristic approach is developed: a Hybrid Granular Tabu Search (HGTS). This algorithm is a tabu search that contains seven different moves to get better solutions and it explores into a granular neighborhood. This granular neighborhood incorporates a term that represents the average length of the arcs in the current solution and a term that represents the remaining route capacity. The algorithm selects the best move among a set of randomly selected moves obtained at each procedure. This algorithm allows no improving moves after $T - 1$ iterations without improving moves (in term of savings). The output of each no-tabu move might be an infeasible solution in both vehicle capacity (but controlled by a parameter) and route leg length. The excess of capacity of the route is fixed in a *capacity correction phase*. Finally, the route leg length is corrected by a *route leg*

Figure 12: Solution of P1(0,0) with Stop Nodes



length correction phase.

From the results, it can be observed that the HGTS outperforms many tested instances with respect to very efficient algorithms with the advantage that our heuristic considers the minimum number of vehicles needed to serve all the demands (as SS does). Thus HGTS gets better improving percentages with respect to TABUSPLIT in 16 instances among all the compared algorithms and 27 instances with respect to SS. When the stop nodes are needed, the algorithm obtains a solution by inserting these nodes to avoid the violation of the maximum route leg length L. We cannot compare our results with other algorithms because this variant of vehicle routing problem is new in the literature.

This paper presents a contribution to the state-of-the-art of the vehicle routing problem due to the introduction of a new variant of this problem. Also, the HGTS obtains the best results for a set of well known instances when the SDVRP is solved. For future research, the properties and particularities of the stop nodes must be studied. Furthermore, because the proposed heuristic represents an upper bound for the best solution of the problem, an algorithm that gets a lower bound could prove useful.

References

[1] D. Aksen, Z. Ozyurt, and N. Aras. Open vehicle routing problem with driver nodes and time deadlines. *Journal of the Operational Research Society*, 58(9):1223–1234, 2006.

[2] R.E. Aleman and R.R. Hill. A tabu search with vocabulary building approach for the vehicle routing problem with split demands. *International Journal of Metaheuristics*, 1(1):55–80, 2010.

[3] R.E. Aleman, X. Zhang, and R.R. Hill. A ring-based diversification scheme for routing problems. *International Journal of Mathematics in Operational Research*, 1(1):163–190, 2009.

[4] R.E. Aleman, X. Zhang, and R.R. Hill. An adaptive memory algorithm for the split delivery vehicle routing problem. *Journal of Heuristics*, 16(3):441–473, 2010.

[5] C. Archetti, M. W. P. Savelsbergh, and M. G. Speranza. An optimization-based heuristic for the split delivery vehicle routing problem. *Transportation Science*, 42(1):22–31, 2008.

[6] C. Archetti, M.W.P. Savelsbergh, and M.G. Speranza. Worst-case analysis for split delivery vehicle routing problems. *Transportation Science*, 40(2):226–234, 2006.

[7] C. Archetti and M. G. Speranza. Vehicle routing in the 1-skip collection problem. *Journal of the Operational Research Society*, 55(7):717–727, 2004.

[8] C. Archetti, M. G. Speranza, and A. Hertz. A tabu search algorithm for the split delivery vehicle routing problem. *Transportation Science*, 40(1):64–73, 2006.

[9] J. M. Belenguer, M. C. Martinez, and E. Mota. A lower bound for the split delivery vehicle routing problem. *Operations Research*, 48(5):801–810, 2000.

[10] P.I. Belfiore et al. Scatter search for a real-life heterogeneous fleet vehicle routing problem with time windows and split deliveries in Brazil. *European Journal of Operational Research*, 199(3):750–758, 2009.

[11] M. Boudia, C. Prins, and M. Reghioui. An effective memetic algorithm with population management for the split delivery vehicle routing problem. *Hybrid Metaheuristics*, pages 16–30, 2007.

[12] V. Campos, A. Corbelan, and E. Mota. A scatter search algorithm for the split delivery vehicle routing problem. In Andreas Fink and Franz Rothlauf, editors, *Advances in Computational Intelligence in Transport, Logistics, and Supply Chain Management*, volume 144 of *Studies in Computational Intelligence*, pages 137–152. Springer Berlin / Heidelberg, 2008.

[13] S. Chen, B. Golden, and E. Wasil. The split delivery vehicle routing problem: Applications, algorithms, test problems, and computational results. *Networks*, 49(4):318–329, 2007.

[14] G. Clarke and J. W. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12(4):568–581, 1964.

[15] GA Croes. A method for solving traveling-salesman problems. *Operations Research*, 6(6):791–812, 1958.

[16] G. B. Dantzig and J. H. Ramser. The truck dispatching problem. *Management Science*, 6(1):80–91, 1959.

[17] G. Desaulniers. Branch-and-price-and-cut for the split-delivery vehicle routing problem with time windows. *Operations Research*, 58(1):179–192, 2010.

[18] M. Dror, G. Laporte, and P. Trudeau. Vehicle routing with split deliveries. *Discrete Appl. Math.*, 50(3):239–254, 1994.

[19] M. Dror and P. Trudeau. Saving by split delivery routing. *Transportation Science*, 23(2):141–145, 1989.

[20] M. Dror and P. Trudeau. Split delivery routing. *Naval Research Logistics*, 37:383–402, 1990.

[21] W. M. Garvin, H. W. Crandall, J. B. John, and R. A. Spellman. Applications of linear programming in the oil industry. *Management Science*, 3(4):407–430, 1957.

[22] B. Garvish and S. Graves. The travelling salesman problem and related problems. Technical report, Working paper 7905. Graduate School of Management, University of Ronchester, Ronchester, NY, 1979.

[23] B. Garvish and S. Graves. Scheduling and routing in transportation and distribution systems: Formulations and new relaxations. Technical report, Graduate School of Management, University of Rochester, NY, 1982.

[24] M. Gendreau, P. Dejax, D Feillet, and C. Gueguen. Vehicle routing problem with time windows and split deliveries. Technical report, 851, Laboratoire d'Informatique d'Avignon, 2006.

[25] M. Gendreau, A. Hertz, and G. Laporte. New insertation and postoptimization procedures for the traveling salesman problem. *Operations Research*, 40(6):1086–1094, 1992.

[26] M. Gendreau, A. Hertz, and G. Laporte. A tabu search heuristic for the vehicle routing problem. *Management Science*, 40:1276–1290, 1994.

[27] F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publisher, 2001.

[28] S.C. Ho and D. Haugland. A tabu search heuristic for the vehicle routing problem with time windows and split deliveries. *Computers & Operations Research*, 31(12):1947–1964, 2004.

[29] M. Jin, K. Liu, and R.O. Bowden. A two-stage algorithm with valid inequalities for the split delivery vehicle routing problem. *International Jounal of Production Economics*, 105(1):228–242, 2007.

[30] M. Jin, K. Liu, and B. Eksioglu. A column generation approach for the split delivery vehicle routing problem. *Operations Research Letters*, 36(2):265–270, 2008.

[31] G. Laporte, Y. Nobert, and M. Desrochers. Optimal routing under capacity and distance restrictions. *Operations Research*, 33(3):1050–1073, 1985.

[32] C. G. Lee, M. A. Epelman, C. C. White III, and Y. A. Bozer. A shortest path approach to the multiple-vehicle routing problem with split pick-ups. *Transportation Research B*, 40(4):265–284, 2006.

[33] P. A. Mullaseril, M. Dror, and J. Leung. Split delivery routing heuristics in livestock feed distribution. *Journal of the Operational Research Society*, 48(2):107–116, 1997.

[34] G. Sierksma and G. Tijssen. Routing heliopters for crew exchanges on off-shore locations. *Annals of Operations Research*, 76(0):261–286, 1998.

[35] P. Toth and D. Vigo. The granular tabu search and its application to the vehicle-routing problem. *INFORMS Journal on Computing*, 15(4):333, 2003.